**AUTOENCODER**

import numpy as np

import tensorflow as tf

from tensorflow.keras.layers import Input, Dense, Flatten, Reshape, Conv2D, Conv2DTranspose

from tensorflow.keras.models import Model

from tensorflow.keras.datasets import mnist

import matplotlib.pyplot as plt

# Load MNIST dataset

(x\_train, \_), (x\_test, \_) = mnist.load\_data()

# Normalize and reshape the data

x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0

x\_train = np.expand\_dims(x\_train, axis=-1)

x\_test = np.expand\_dims(x\_test, axis=-1)

# Add noise to the data

noise\_factor = 0.5

x\_train\_noisy = x\_train + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=x\_train.shape)

x\_test\_noisy = x\_test + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=x\_test.shape)

x\_train\_noisy = np.clip(x\_train\_noisy, 0.0, 1.0)

x\_test\_noisy = np.clip(x\_test\_noisy, 0.0, 1.0)

# Build the autoencoder model

input\_img = Input(shape=(28, 28, 1))

# Encoder

x = Conv2D(32, (3, 3), activation='relu', padding='same')(input\_img)

x = tf.keras.layers.MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)

encoded = tf.keras.layers.MaxPooling2D((2, 2), padding='same')(x) #low dimensional image with noise created here

# Decoder

x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)

x = tf.keras.layers.UpSampling2D((2, 2))(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)

x = tf.keras.layers.UpSampling2D((2, 2))(x)

decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x) # image reconstructed without noise

# Autoencoder

autoencoder = Model(input\_img, decoded) # training with the original input and reconstructed output.(samples of 60k for mnist dataset)

autoencoder.compile(optimizer='adam', loss='binary\_crossentropy')

# Train the autoencoder

autoencoder.fit(x\_train\_noisy, x\_train,

epochs=10,

batch\_size=128,

shuffle=True,

validation\_data=(x\_test\_noisy, x\_test))

# Predict on noisy images

denoised\_images = autoencoder.predict(x\_test\_noisy)

# Display the results

n = 10

plt.figure(figsize=(20, 4))

for i in range(n):

# Display original clean images

ax = plt.subplot(3, n, i + 1 + 2 \* n)

plt.imshow(x\_test[i].reshape(28, 28), cmap="gray")

plt.title("Original")

plt.axis("off")

# Display original noisy images

ax = plt.subplot(3, n, i + 1)

plt.imshow(x\_test\_noisy[i].reshape(28, 28), cmap="gray")

plt.title("Noisy")

plt.axis("off")

# Display denoised images

ax = plt.subplot(3, n, i + 1 + n)

plt.imshow(denoised\_images[i].reshape(28, 28), cmap="gray")

plt.title("Denoised")

plt.axis("off")

plt.show()

# Plot training and validation loss

history = autoencoder.history

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Loss vs. Epochs')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()